CS 111 Midterm Exam

Nate Nateerawut Sookwongse

TOTAL POINTS

91 / 100

QUESTION 1

1 Pages and page frames 10 / 10

✓ **- 0 Correct**

- 10 No answer.
- 2 Did not explain that pages get placed into page frame.
- 1 Said page is mapped to page frame
- 1 No mention of location of page & page frame in system
- 4 Incorrect Reasoning
- 3 Not accurate enough
- 1 No mention of virtual address
- 5 Page frame contains exactly one page.
- 2 Did not mention that page and page frame are the same size.

QUESTION 2

2 ABIs 10 / 10

✓ **- 0 Correct**

- 10 No answer
- 1 Did not mention operating system
- 4 Said applications don't need to worry about hardware differences
- 2 Off-topic
- 3 Incorrect reasoning
- 2 Did not mention software distribution
- 3 Did not mention hardware and OS
- 2 Left out hardware

QUESTION 3

3 Information hiding 7 / 10

- 0 Correct
- 10 No answer.
- 6 Primary benefit is to avoid bugs arising from improper dependencies among modules.

✓ **- 3 Major element of benefit is that it allows changes in module implementation.**

- 7 Really about hiding details of OS modules' implementation, not about concealing processes' address spaces from each other.
- 2 Nothing to do with open vs. closed source.
- 9 Primarily an issue involving abstraction.

QUESTION 4

4 Context switches 7 / 10

- 0 Correct
- 2 Not mentioning general registers
- 2 Not mentioning PC

✓ **- 2 Not mentioning Stack ptr**

✓ **- 1 Not mentioning PSW**

- 2 No discussion of memory mapping data
- 2 No need to explicitly save data, since it's already sitting in memory.
- 3 Generally nothing goes to disk on a context switch.
- 1 What about memory needs to be saved?
- 2 Much of this stuff need not be saved, since it's already in memory. OS just needs to be sure it can be found again when process is switched back in.
- 2 The PCB is an OS data structure that exists as long as the process is around, so it need not be saved on a context switch.
- 1 File size has nothing to do with a context switch.
- 1 I have no idea what the flag you're talking about is.
- 1 "state of the process" is vague.
- 2 Caches aren't saved.
- 2 No need to update a file descriptor during a context switch.

QUESTION 5

5 Trap tables 10 / 10

✓ - 0 **Correct**

- 10 No answer
- 3 Answer incomplete, should mention trap table is used to specify what code to run when trap occurs.
- 8 Wrong answer.
- 3 Answer incomplete.
- 2 User process has no thing to do with trap?

QUESTION 6

6 Race conditions 10 / 10

✓ - 0 **Correct**

- 10 No answer
- 5 Answer incomplete.
- 8 Answer incorrect.
- 2 Missing some details.

QUESTION 7

7 Blocking and threads 10 / 10

✓ - 0 **Correct**

- 10 No answer or Wrong anwser
- 5 Missing: User-mode threads block other threads of the same process.
- 5 Missing: Kernel-mode threads do not block other threads of the same process, as other threads can be scheduled to run on the same or another core.

QUESTION 8

8 STCF 10 / 10

✓ - 0 **Correct**

- 10 No answer or Wrong answer
- 5 Missing: interrupt the running one OR switch to the newly-added shorter ones.

QUESTION 9

9 Fork and exec 7 / 10

- 0 Correct
- 10 No answer
- 4 Not mentioning code replacement.
- 3 Not mentioning stack replacement.

✓ - 3 **Not mentioning heap replacement.**

- 9 The question was about what happens after the exec, not the fork.
- 8 What resources are replaced by the exec?
- 7 Stack and code are changed by exec.
- 5 Fork/exec work with processes, not threads.
- 2 Even any data written after fork gets replaced by exec.
- 2 The old stack is totally overwritten.
- 10 Totally wrong. Nothing to do with multithreading and multicore.
- 6 So, what resources are replaced?

QUESTION 10

10 Fragmentation for memory management schemes 10 / 10

✓ - 0 **Correct**

- 10 No answer
- 5 Not identifying internal fragmentation for pages.
- 5 Paged segments suffer 1/2 page fragmentation.
- 5 Fixed segments suffer 1/2 internal segment fragmentation.
- 3 On average 50%
- 2 The 1.5% was a particular example. It will be 1/2 page, on average.
- 2 Internal fragmentation has little to do with how long the system runs, unlike external.
- 2 Paging and fixed size partitions never experience external fragmentation.
- 2 The paging form of fragmentation you describe is internal fragmentation.
- 2 Paging doesn't use binary buddy. It allocates in fixed size pages.
- 4 Fixed segments are likely to waste more memory on internal fragmentation than paging, not less.
- 3 No external fragmentation with paging.
- 5 No answer on segmented system.
- 2 Calling internal fragmentation "external".
- 1 This form of fragmentation is called "internal."
- 4 Paged segment fragmentation only occurs in the last page.

Midterm Exam
CS 111, Principles of Operating Systems
Fall 2017

Name: Nate Sookwongse

Student ID Number: 204 645 821

This is a closed book, closed note test. Answer all questions.

Each question should be answered in 2-5 sentences. DO NOT simply write everything you remember about the topic of the question. Answer the question that was asked. Extraneous information not related to the answer to the question will not improve your grade and may make it difficult to determine if the pertinent part of your answer is correct. Confine your answers to the space directly below each question. Only text in this space will be graded. No question requires a longer answer than the space provided.

1. In a system using modern virtual memory techniques, what is the relationship between a page and a page frame?

A page is a small (e.g. 1 or 4 kB), fixed-sized allocation of the virtual address space.

A virtual page can be mapped to a physical page frame in memory (via a page table / TLB).

This virtualization of pages and page frames gives the illusion that a process has a large, contiguous address space, when really adjacent pages may be mapped to page frames throughout memory.

2. Why are operating system ABIs of importance for convenient application software distribution?

ABI's are important because these specifications are what guarantee that a software is compatible and can run correctly.

Software is built to comply with a particular ABI so that they know the program will work for consumers with that particular OS and hardware.

Poorly defined ABI and/or choosing not to follow an ABI can lead to software bugs and incompatibilities (especially at OS updates)

3. Why is information hiding a good property in an operating system interface?

It is good to hide information in an OS interface to encapsulate the complexity and increase ease of use. It's not so important to know how or what's actually happening behind the scenes. What's important is that an interface is clearly defined so that others know what's the purpose and when to use it.

4. When an operating system performs a context switch between processes, what information must the OS save?

To perform a context switch, an OS must save the current process state, so that it can restore the state back to original when it comes time to run this process again.

A process state contains information such as the process's registers, program counter (PC), address space (page table)

5. What is the purpose of a trap table?

The purpose of a trap table is to properly handle any traps/exceptions that may arise during program execution.
A trap table contains pointers to particular trap handlers in the hardware, which can be indexed when a trap is called (e.g. a system call).

6. What is a race condition?

A race condition exists when the outcome of a program execution may differ, depending on the particular execution times within the program. (also known as being indeterminate).

A race condition may be an issue in the case of multi-threaded applications, where the outcome may be dependant on when each thread is scheduled.

with critical section(s) of code attempting to write to a shared resource

7. Why is blocking a problem for user-mode threads? Why isn't it a problem for kernel-mode threads?

The OS cannot see these multiple user-mode threads and instead treats multiple user-mode threads as a single process. Therefore, when blocking is used in a user-mode thread, no other thread can run, because the process as a whole is being blocked.

This is not an issue for kernel-mode threads, which can utilize the multiple cores of the system (assuming there are multiple cores), and simply schedule a different thread to run on another core.

8. Why does Shortest Time-To-Completion First (STCF) scheduling require preemption?

New processes may be continually added to the queue. Thus, preemption is necessary so that a process that may currently be running can be paused if there is a newer, shorter job that is added to the queue, and allow this shorter job to be scheduled instead. Without preemption, STCF would only properly work if all the jobs were given initially, which is not typical or realistic.

9. When a Unix-system follows a fork with an exec, what resources of the forked process are replaced?

The stack, registers, and code for the forked process process is replaced and initialized for the new program to be executed. After the exec call, it is as if the forked process never even happened.

If copy-on-write is used, this is simple to be done, with fewer resources to replace

10. What form of fragmentation do we still suffer if we use a paging memory management system? For a segmented paging system, how much fragmentation per segment do we see? fixed-sized segments

Although a paging system helps prevent external fragmentation, it may still suffer from internal fragmentation. Because pages and page frames are allocated in fixed-sized segments, a process may not require and utilize the entire allocated segment.

On average, a fixed-sized segment may suffer from 50% internal fragmentation. However, if a process requires multiple pages, then only the last page (segment) might suffer from fragmentation, and overall, less than 50% fragmentation.